Mobile Gateway for Wireless Sensor Networks utilizing drones

Ioan Deaconu, Andrei Voinescu
Automatic Control and Computers Faculty
University Politehnica of Bucharest,
{ioan.deaconu@cti.pub.ro, andrei.voinescu@cs.pub.ro}

***Abstract*—Mobile Gateway for Wireless Sensor Networks utilizing drones**

The use of Wireless Sensor Networks is increasing and are beeing used in places were maintenance is dificult. Because of the power limitations, the nodes have a transmision range of only a few tens of meters which results in creating a multi-hop wireless sensor network architecture in order to send the data from any node to a gateway. But in some cases, creating this type of network might be to difficult or to expensive. Our design offers a truly mobile gateway using a SparrowDongle connected to a Parrot Drone. This setup allows us not only to collect data from a node, but also to debug a wireless sensor network. [4]

***Index Terms*—gateway, wireless sensor networks, drone**

Figure 1. SparrowDongle connected to AR Parrot Drone 2.0

I. Introduction

Wireless Sensor Networks are being used more and more in almost every field imaginable in order to collect and improve our life, fileds like home automation, agriculture, military, space exploration etc. In order to collect the data, gateway platforms [5], [2], [3] are required and most of them are stationary bulky devices or PCs connected to one of the wireless nodes that serve as a base-station. We aim to show in this paper that there exists a solution for a truly mobile gateway design that can be used either for collecting data or for debuging large wireless network infrastructures.

We have conected to an AR Parrot Drone 2.0 our SparrowDongle, a USB stick featuring two microcontrollers that can connect to 2.4GHz Zigbee nodes or to our own node design Sparrowv3.2. We will show an overview of the system architecture in Chapter II, the hardware and software implementation in Chapter III and results of using the gateway in Chapter IV.

II. System architecture

The canonical implementation for Atmel Zigbee transceivers is a USB stick device with a USB controller and radio transceiver, which means that the USB controller unit (the only controller present) on the device is responsible for both radio and USB stack communication. This lack of separation between key functions of the gateway platform leads to the undesirable effect of software for the USB stack competing for MCU time with the wireless stack. This greatly limits both the USB throughput, features of the device and the complexity of the wireless stack. The wireless stack in this scenario cannot have tight timings built around receiving and transmitting packets on the wireless link, as transfer to and from the microcontroller unit (MCU) is both slow and delayed by (possible) USB tasks running asynchronously.

Our implementation differs in that respect by including two separate controllers on the gateway device, one for USB communication and the other for the 2.4Ghz Zigbee stack. This approach has several key advantages, described in sections II-A and II-B.

Additionally, a number of design features were included for ease-of-use in research and development, outlined in II-C, II-D

A. Separation of functionality

USB communication is poll-based and initiated by the USB host. The SparrowDongle stick acts as a USB device and its role requires frequent (every millisecond) and low-latency communication with the USB host. Having two controllers, the RF controller can run any RF communication stack without having the USB code intrude on key timings. The serial link that connects the two controllers has both sufficient speed and simplicity to allow each controller to dedicate most computing cycles to handling communication on the USB and wireless links, respectively.

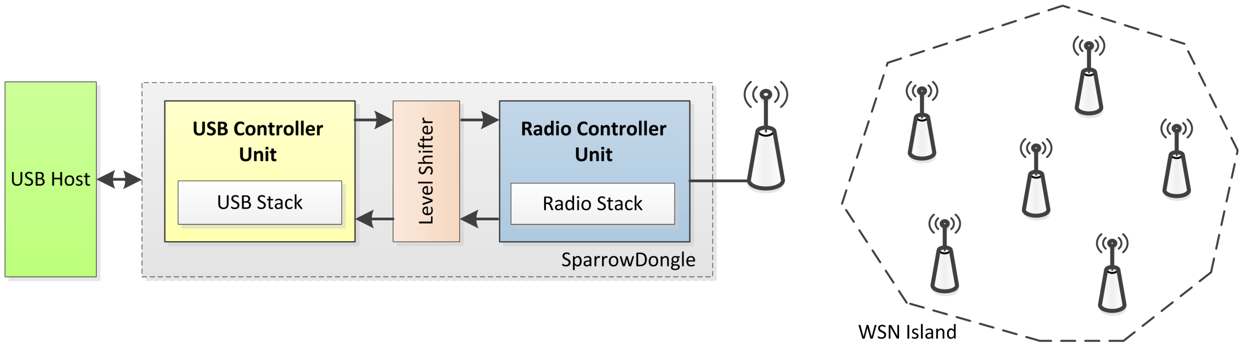

Figure 2. SparrowDongle stick architecture

B. Homogeneity

The components used for the RF communication in the SparrowDongle stick are identical to those used for the Sparrowv3.2 nodes, thus they can share the same codebase (As opposed to having an implementation for 8-bit AVR and an implementation for ARM/x86). In many cases, gateways run on different architectures than wireless sensor nodes and require stacks re-purposed for that specific architecture (or for that platform, in the case of the canonical Atmel implementation of the gateway). SparrowDongle eliminates the need for a different branch of the same wireless stack since the code running on the gateway is virtually identical to that running on any sensor node.

C. Ease of programming

SparrowDongle offers an easy-to-use programming and debugging interface for both the USB controller and the RF controller. To keep the overall size of the device small, we used shared ISP (In-System-Programming) and JTAG (Joint Test Action Group) headers for programming the two controllers. This only requires an ISP header, a JTAG header and two headers for jumper selection (One jumper connects the ISP signal SCK to the respective SCK signal pin on one of the controllers, the other connects the JTAG signal TCK to its respective signal pins on one of the controllers), as shown in Figure II-C.

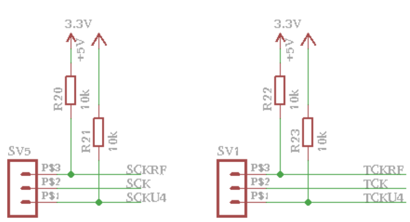

Figure 3. Header selection for programming clock signals

D. External Antenna

The design for the SparrowDongle wireless stick includes an optional UFL connector for an external antenna, which greatly increases range. A large, 8dBi omni-directional antenna mounted on both gateways and nodes would amount to around 200 metres of communication range, well over the 70m measured with the default antennas.

III. Implementation

A. Hardware Details

The hardware components used are the following:

- a SparrowDongle with two microcontrollers: The 8-bit ATMega128RFA1 which has an on-chip 2.4GHz wireless transceiver and the ATMega32U4, both from Atmel.
- a SparrowV3.2 with an ATMega128RFA1 microcontroller
- a AR Parrot Drone v2.0
- an laptop or mobile phone with android/ios for controlling the drone

Because of the addition of SparrowDongle, the drone's poliester hull has been carved a little in order to accomodate it.

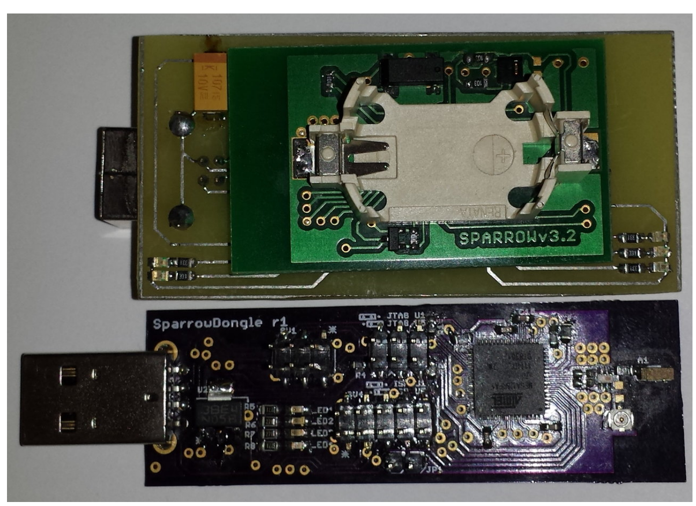

Figure 4. SparrowV3.2 and SparrowDongle

B. Software Implementation

The SparrowDongle gateway dumps every data received on the serial. It is always in a listen for data state. When it receives the data, it sends back an ack to let the SparrowV3.2 know that it can begin sending the entire data to the mobile gateway.

The SparrowV3.2 node is sending periodicaly a small data packet to check if a gateway is available. When it receives the ack for the packet it starts sending the stored data to the gateway. The data sent can vary, from sensor readings, to debuging informations in order to check the state of the Wireless Sensor Network.

The data gattered by the gateway is saved into the AR Parrot Drone's internal memory. The data can be accesed at any time by any device connected to the drone's wireless network via ftp.

IV. Results

In this chapter we will cover the results obtained with this gateway platform and other possible applications.

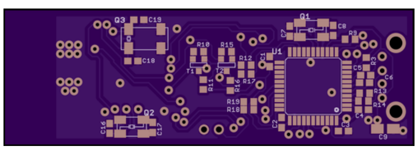

Figure 5. Bottom side of SparrowDongle PCB

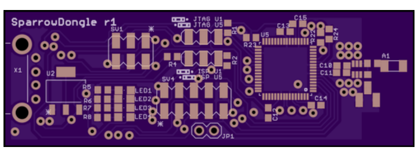

Figure 6. Top side of SparrowDongle PCB

A. Performance

Throughput testing was done with back-to-back packets sent at 250kbps over 2.4GHz with one sending node in acceptable range, with no losses. This is due to the double-buffering used in receiving packets from the wireless network. As soon as one packet ends, a signal is sent to the radio controller unit with a small delay of $9\mu S$. Even if a new packet starts in that small interval, receipt of the new packet goes unhindered as the first bytes of the old packet have already been transfered to a different memory location from which they will be sent to the USB controller unit via the serial connection.

B. Software configurations

A great advantage of using a dedicated USB Controller Unit for the gateway is that it can be programmed as one of several USB Communication Classes. The USB Controller Unit is not limited in implementing any of these classes since most of the computing power at its disposal is reserved for USB. SparrowDongle can appear as different USB devices:

- *Virtual Serial Port*: Communication with the wireless island around the gateway can be made via a serial link, incoming packets will appear on the receive end of this port and packets will be sent on the transmit end. In typical Unix fashion, our implementation sends packet in ASCII for ease of use and debugging. They are converted to binary form on the Radio Controller Unit of the gateway
- *Ethernet Emulation*: In this fashion, packets are received on the gateway and then encapsulated in an Ethernet packet sent over the USB link (Ethernet is emulated between the USB device and USB host)
- *Network card*: SparrowDongle behaves as a wireless network card, the operating system will register a network interface for the gateway and addresses assigned to this interface will change the gateway's address in the wireless medium (as opposed to changing the address for the emulated Ethernet)
- *Mass Storage*: SparrowDongle can offer a virtual filesystem interface for innovative data acquisition from the wireless sensor network. In accordance with the Unix philosophy of "everything is a file", the virtual filesystem offered by the USB stick could have a file for each wireless node where it stores recent data (as much as the gateway can store in its volatile memory, 1-2 records per node). The software implementation for this interface is under development.

C. Applications

The versatility of the SparrowDongle gateway platform allows it to be deployed in a wide range of applications, whether the gateway has to be connected to a PC or a small embedded device, whether it has to implement a virtual serial connection or to emulate an ethernet link.

For instance, these are the application in which SparrowDongle is currently deployed:

- Connected to a Windows PC, feeding wireless sensor data into a service framework for building control, in the FCINT project. [1]
- Connected to an Embedded Linux board, such as the small RaspberryPi, for plug-and-play monitoring of a wireless sensor island.
- Connected to a Parrot Drone [6], for remote monitoring using a mobile gateway.

V. Conclusion

The paper presented a versatile gateway platform for wireless sensor networks that is both capable of serving current application needs as well as offer the ability to interface sensor networks in a novel way (sensor data as files).

The platform is easy-to-use and to program due to clear separation of communication mediums on different controllers and benefits from the elimination of code duplication in the case of the radio controller unit.

ACKNOWLEDGMENT

The research presented in this paper was supported by the EU POS-CCE project “Ontology-based Service Composition Framework for Syndicating Building Intelligence: FCINT” No.181/18.06.2010

REFERENCES

[1] Fcint - service composition framework based on onthologies for knowledge and information aggregation in smart buildings. *Available: fcint. ro*, 2013.

[2] B. da Silva Campos, J. J. Rodrigues, L. D. Mendes, E. F. Nakamura, and C. M. S. Figueiredo. Design and construction of wireless sensor network gateway with ipv4/ipv6 support. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5. IEEE, 2011.

[3] B. da Silva Campos, J. J. Rodrigues, L. D. Mendes, E. F. Nakamura, and C. M. S. Figueiredo. Design and construction of wireless sensor network gateway with ipv4/ipv6 support. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5. IEEE, 2011.

[4] I. Doroftei, V. Grosu, and V. Spinu. Omnidirectional mobile robot–design and implementation. *Habib, Maki. Bioinspiration and Robotics, Walking and Climbing Robots. Viena, Austria: I-Tech Education and Publishing*, pages 511–528, 2007.

[5] J. Hill, M. Horton, R. Kling, and L. Krishnamurthy. The platforms enabling wireless sensor networks. *Communications of the ACM*, 47(6):41–46, 2004.

[6] A. Parrot. Drone. *Available: ardrone. parrot. com*, 75, 2012.